NOLTR 61-114

THE PERCEPTRON AS AN ADAPTIVE CLASSIFICATION DEVICE

26 OCTOBER 1961

NOL
UNITED STATES NAVAL ORDNANCE LABORATORY, WHITE OAK, MARYLAND

NOLTR 61-114

THE PERCEPTRON AS AN ADAPTIVE CLASSIFICATION DEVICE

Prepared by:

C. Nicholas Pryor, Jr.

ABSTRACT: Many data reduction problems fall under the general category of classification, such as sonar target classification or character recognition. These may be represented as a process of mapping a hyperspace representing the input data into an output space representing distinct decisions or categories. The majority of practical problems may be simplified by the assumption of continuity in the mapping process, and it is desirable to mechanize the classification problem according to this assumption.

A class of adaptive mechanisms known as learning machines has the capability of classifying input data, and a machine of this sort can adjust itself to satisfy the desired classifying criterion if it is given a collection of input data identified as to its desired class. One machine of this type is known as the Perceptron and is discussed in some detail. A simpler form of the machine, the half-Perceptron, is also discussed.

These two types of learning machines were simulated on the IBM 704 computer, and sample problems in classification were processed on them. The results demonstrate some of the capabilities and limitations of the two machines.

A discussion is also included on the possible future role of learning machines, and on their application to naval problems.

PUBLISHED NOVEMBER 1961

U. S. NAVAL ORDNANCE LABORATORY
White Oak, Silver Spring, Maryland

NOLTR 61-114 26 October 1961

This report gives the results of programming an IBM 704 Computer to act as a PERCEPTRON (a self-adaptive or learning machine) on two simplified classification problems as a preliminary to a study of submarine classification with sonar data. The work on this project was supported under Task No. RUSD-4C150, PUFFS Technical Direction. The report is for the information of other scientists interested in classification problems or in the use of computers as adaptive machines.

W. D. COLEMAN
Captain, USN
Commander

[illegible]

Z. I. SLAWSKY
By direction

CONTENTS

ILLUSTRATIONS

REFERENCES

1. W. Ross Ashby, Design for a Brain, Wiley and Sons, New York, 1954.

2. John Von Neumann, The Computer and the Brain, Yale University Press, New Haven, 1958.

3. Frank Rosenblatt, The Perceptron - A Theory of Statistical Separability in Cognitive Systems, Cornell Aeronautical Laboratory, Inc., Report No. VG-1196-G-1, January 1958.

4. Frank Rosenblatt, The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Psychological Review, 65, 1958, pp. 386-408.

5. Frank Rosenblatt, Two Theorems of Statistical Separability in the Perceptron, CAL Report No. VG-1196-G-2, September 1958.

6. Frank Rosenblatt, Analysis of Very Large Perceptrons in a Finite Universe, CAL Project PARA Technical Memorandum No. 2, October 1958.

7. Frank Rosenblatt, Perceptron Simulation Experiments (Project PARA), CAL Report No. VG-1196-G-3, June 1959.

8. Frank Rosenblatt, On the Convergence of Reinforcement Procedures in Simple Perceptrons, CAL Report No. VG-1196-G-4, February 1960.

9. Albert E. Murray, A Half-Perceptron Pattern Filter, CAL Project PARA Technical Memorandum No. 16, July 1960.

10. R. D. Joseph, On Predicting Perceptron Performance, IRE International Convention Record, 1960 (Part 2).

*11. Frank Rosenblatt, Principles of Neurodynamics, CAL Report No. VG-1196-G-8, 15 March 1961.

*This reference contains a review of the entire Perceptron concept and theoretical performance, as well as a very complete bibliography on the subject.

THE PERCEPTRON AS AN ADAPTIVE CLASSIFICATION DEVICE

THE CLASSIFICATION PROBLEM

1.1 In many problems in data reduction such as character recognition and sonar target classification, a large amount of data is available (such as light intensity at many points on a retina or spectral intensity at many frequencies); and the problem is to identify a given set of input data as belonging to one of several possible disjoint classes or categories of input signal, each of which must give rise to a distinct decision or output state of the data reduction system. If there are N pieces of input data, each with a number of discrete or a range of continuous values, the input to the system may be represented by a point in an N-dimensional "input space". Correspondingly if there are M possible output decisions that may be made (that is, M possible output classes), we may think of the output as a one-dimensional space containing M discrete points; and the problem becomes one of a many-to-one transformation of points from the input space to the output space. Another way of considering the problem is that of labeling every point in the input space with the identification (perhaps a number from 1 to M) of the output state appropriate to it.

1.2 So far we have assumed that the output state required for a given point in input space is independent of that required for any other point, and that it is necessary to examine each point in detail to determine its required output. This is an overwhelming job in any reasonable problem and of course impossible if any one of the input variables is continuous. However, this is the approach which must be taken if any of the usual switching circuit or logical computer realizations is to be attempted.

1.3 In most problems of practical interest it is reasonable to assume that small changes in one or more of the input variables will tend not to change the desired output state. That is, nearby points in the input space will tend to belong to the same class, and the transformation from input space to output space will tend to be continuous. The problem now becomes a simpler one of determining the boundaries separating regions of input space belonging to different classes. It is no longer necessary to study every possible input state, but only enough to estimate these boundaries in the input space. New points (that is points which were not initially used in establishing the boundaries) are then categorised according to the region in which they fall, which means that they are assumed to belong to the same class as neighboring points. If this decision proves to be wrong, it is then necessary to adjust the boundaries or perhaps to form a new region to take the new point into account.

1.4 In spite of this simplification, determining the proper boundaries from a collection of input data and desired outputs is still an enormous job (made even more so by the difficulty of thinking in more than three dimensions) and it becomes desirable to mechanize this operation. In general this problem is handled by some sort of adaptive mechanism to which a sample of input data is fed, along with information concerning the desired output. Internally, the machine consists of a large logical network, whose logical function may be varied by adjustment of some parameters of the network. Certain types of logical systems have the desired characteristic of tending to produce the same output for similar inputs, and these networks may be thought of as defining the regions of input space for which a given output will occur. If the machine output differs from the desired output, a corrective signal is fed back to the machine. This performance feedback is designed to adjust the internal construction of the logical network in such a way as to correct, or reduce the probability of, the wrong answer. The effect of this, in a properly designed mechanism, is to move the boundaries between regions of input space in such a way that the given input point will be included in the proper region.

1.5 In an ideally designed machine, only the boundary in question will be moved; and training for one input point will not affect the response for other points outside its immediate vicinity. It is usually not possible to attain this degree of independence in a practical mechanism, and all boundaries will tend to move during a given training operation. This reduces the convergence to the proper configuration to an iterative procedure of successive approximations, and it is usually necessary to go through a given finite list of input data a number of times before error-free operation can be obtained. This gives the machine a statistical appearance, even though the internal operation is usually not randomly determined, and the size of the machine generally makes anything other than statistical analysis impractical. For this reason, it is usually sufficient in the training procedure simply to make corrections in the direction of including the input point in the desired region rather than necessarily forcing the machine to produce the desired answer before going on to the next input sample.

1.6 The complexity of the internal logical network of the adaptive system determines the degree of detail in the apportioning of input space which may be stored in the memory of the logical network. Where the input space is divided into many separate regions with complicated boundaries, a very elaborate switching circuit would be required to realize the logical functions implied by these regions. Thus it would be expected that a rather large adaptive logical network might be required to simulate this operation. Conversely, certain simple geometries of the regions in input space may be realized using rather small logical networks. In addition to this, many of the simple logical systems which may be used in an adaptive learning machine have fundamental limitations which prevent them from being able to realize certain geometries within the input space. These weaknesses generally become apparent upon careful study of the philosophy used in the machine organization.

1.7 The following sections of this report discuss the organization of two types of adaptive classification or learning machines. Their simulation on an IBM 704 computer is also discussed, and sample problems are shown which demonstrate the capabilities and characteristics of the machines further.

THE PERCEPTRON

2.1 A device which seems rather well suited for the classification problem is the Perceptron, developed at the Cornell Aeronautical Laboratory. The Perceptron was developed as a model of the possible construction of the human nervous system, and thus its components may be identified with the receptor neurons, motor neurons, and intermediate "gray matter" neurons in the nervous system. The organization of the device is shown in Figure 1. Inputs to the system are represented by the states of the Stimulus (S) Units, and through the logical network the excitations to the Response (R) Units are determined. The binary state of the several R Units is considered to be the output of the device.

2.2 Th logical function is carried out by the intermediate Association (Units, which act as the connections between the S and R Units. Each A Unit has a number of input connections (dendrites) which are randomly connected to several of the S Units. The output of the A Unit is connected at random to an input of one of the R Units. Of course, no generality is lost by reordering the A Units so that all those connected to a given R Unit are located together as shown in Figure 1. Each A Unit is a two-state device which is either on or off depending on whether the total excitation to the A Unit (sum of all its inputs) exceeds a given threshold level. If the A Unit is on, a signal proportional to its "Value" is placed on its output terminal. If it is off, either no signal or minus the "Value" is placed on the output, depending on the construction of the particular machine. Each R Unit then becomes either on or off depending on whether its input excitation (the sum of the outputs of all A Units connected to it) is above or below a fixed threshold.

2.3 In the nervous system the output of each neuron is a signal of fixed magnitude, and the triggering level or threshold of the neuron varies according to some electrochemical process within the neuron. Learning, or modification of the logical network within the nervous system, is accomplished by varying the triggering level of the individual neurons according to the reinforcement given to certain responses and to whether the neurons were active during the response. In the Perceptron, the thresholds of the individual A Units are kept fixed, but their Values are varied according to the reinforcement given the machine. The two methods of learning are roughly equivalent, but in the Perceptron with its smaller number of components (roughly 10^3 A Units against 10^{10} neurons in the brain) finer detail of learning may be accomplished by varying the Values and keeping the thresholds fixed. Of course it is necessary to adjust the Values during the training operation in some systematic way that will tend to converge toward solution of the problem. A number of procedures for adjusting the A Unit Values have been studied and their characteristics reported by Cornell Aeronautical Laboratory.

2.4 Each of the A Units may be thought of as a basic logical element which performs a boolean operation on its inputs. In particular this operation is the symmetric function $S_{T,\ldots,N}$ where there are N inputs to the A Unit and the threshold is T. The inputs to the A Unit may be restricted to the outputs of the S Units, or for a more general logic both the S Unit outputs and their complements may be made available as inputs for the A Units. The excitation to the R Units is then some linear combination of the symmetric functions realized by the A Units, and it is this linear combination that is varied during the learning process. Finally the greater-than- or less-than decision in the R Unit produces the last stage of logic. Thus the overall machine looks like a logical network, although it is really composed of two nonlinear elements separated by a linear operation. In general the number of possible logical functions which the machine can realize is limited by the complexity of the logical system, or by the number of A Units in the Perceptron. Thus the capability of a Perceptron for discovering and learning complex input-output relationships is roughly determined by the number of A Units in the machine, and the performance of a Perceptron will be improved by increasing the number of A Units.

2.5 There is, of course, a maximum number of A Units which may be used without duplication, due to the finite number of possible combinations of wiring between the S Units and the A Units. If there are N Stimulus Inputs, then there are $\frac{N!}{n!(N-n)!}$ ways in which n of them can be connected to an A Unit. If each of the connections can either be to the direct output of the S Unit or to its complemented output, then the number of distinct connections is increased by a factor of 2^n. Summing these overall values of n from 0 to N, we obtain 2^N possible ways of wiring A Units if complements of the S Unit outputs are not used and 3^N ways if complements are allowed. However, for each A Unit with n input connections, there are n possible settings of its threshold which will give distinct logical operation. Thus n different A Units could be built with the same input connections but differing in thresholds, which will give different logical outputs. It must be noted that a given A Unit with n inputs and a threshold of k will have a logical output complementary to that of another A Unit with its n inputs connected to the complements of the same set of n S Units and with a threshold of n-k. Half the A Units will then have outputs complementary to the other half, so there are only $\frac{n}{2}\, 2^n \frac{N!}{n!\,(N-n)!}$ logically independent A Units possible with n inputs from the set of N Stimulus Units. Summing this over n from 0 to N yields $\frac{N}{3}\, 3^N$ for the total number of different A Units which may be built into an N input Perceptron. The number of wires involved in the connections from the S Units to the A Units may also be computed since there are n wires to each A Unit with n inputs. Thus the number of wires in the S - A matrix is $\sum_{0}^{N} \frac{n^2}{2}\, 2^n \frac{N!}{n!\,(N-n)!}$ or $\frac{N}{9}\,(2N+1)\cdot 3^N$.

2.6 The size of these numbers can best be appreciated by considering a simple example. In a 10 input machine there would be 1024 possible A Units if no complements were allowed or 59,049 A Units if S Unit outputs and their complements were allowed, using fixed thresholds in both cases. If all logically distinct thresholds were allowed in order to produce the most general machine, there could be as many as 196,830 A Units used without duplicating or complementing any logical functions. This machine would require 1,377,810 wires in the connections between S Units and A Units. While these numbers are still very small compared to the $2^{(2^N)}$ or approximately 10^{308} different logical functions which it is possible to form from the 10 inputs, it is fairly obvious that one would not want to build the most general Perceptron for even ten inputs using present circuit techniques. Thus it is necessary to build Perceptrons using only a small percentage of their total possible A Units, generally limiting the selection to some class (such as a fixed value of n) and then taking only a random sampling of that class. While some work has been done on the slope and asymptote of the learning curves of Perceptrons as a function of the number of A Units, it is still not clear just what limits are placed on the apportioning of the input space by a Perceptron with a given number of A Units.

HALF-PERCEPTRON

3.1 A simpler form of learning machine which is much smaller than the Perceptron has also been proposed, and it is termed a half-Perceptron because it is really a degenerate form of the Perceptron. If a Perceptron is built with exactly N Association Units connected to each of the R Units, and each of the A Units is connected simply to one of the S Units, the machine reduces to the form of Figure 2. In this device there is clearly only one logical transormation from the input space (states of the S Units) to the output space (excitations of the R Units), while in the Perceptron there are two layers of logical transformation. The first of these may be considered as a transformation from input space to A Unit excitation space, and the second as a transformation from A Unit space to the output space. The additional logical "depth" in the Perceptron allows it to realize much more complex logical functions than can the half-Perceptron with a logical depth of only one. It seems reasonable to expect that a machine similar to the Perceptron with a logical depth of three or more (obtained by two or more layers of A Units, with each layer providing the excitation for the next) would be even more powerful than the Perceptron.

3.2 The half-Perceptron with a single R Unit is a sufficiently simple machine that it is not too difficult to see the restrictions on the functions that may be realized by it. Basically the device produces an arbitrary linear combination of the input excitations and then determines whether this is greater or less than a fixed threshold. In terms of the partitioning of the input space, the result is that the space will consist of only two regions or decision classes; and these will be separated by an arbitrarily placed and oriented N-dimensional hyperplane. This restricts the types of functions which may be realized by the half-Perceptron, and the restriction grows more severe as the number of inputs increases. For example both of the two functions of no inputs, all of the four functions of one variable, and 14 of the 16 functions of two variables may be realized; while only 104 of the 256 functions of three variables may be realized. In the two-input case such functions as the logical "or" or "and" may be realized, while the disjunctive functions testing for agreement or disagreement in sign cannot be realized.

3.3 In half-Perceptrons with more than one R Unit, the same restrictions apply to each of the R Units individually. However, if the several R Unit outputs are taken together in some logical fashion to represent a single decision (corresponding to adding a second layer of logic), more complicated regions can be formed although they will still be bounded by hyperplanes. Thus, for example, the test for agreement of two or more inputs may be made as the logical "or" of two R Unit outputs; one of which produces the "and" of the required inputs and the other of which produces the "not-or" of the inputs.

COMPUTER SIMULATION OF THE PERCEPTRON

4.1 The IBM 704 computer simulation program for the Perceptron is given in Appendix 1. The program was written in SHARE symbolic language rather than in FORTRAN in order to provide more flexibility in coding for decreased running time. Running of the program is intended to be under control of the BELL Operating System, and the modified FORTRAN input and output routines available as part of the BELL system were used for all input and output routines except for binary tape operations. Because of BELL system use of sense switches, program stops were necessary within the main program in order to allow resetting of the sense switches after leaving and before re-entering BELL system control.

4.2 The number of S Units for the Perceptron was chosen to be 64, and the number of R Units can be varied from one to six as determined by the initial parts of the Perceptron program. The number of A Units was then selected to be as large as possible, consistent with machine size and operating speed. The most critical part of the program for speed considerations is the computation of the A Unit excitations from the S Unit states since this requires a determination of all the S to A connections for each input data sample. Because of the large number of connections this must be done in the fastest way possible, even at the sacrifice of storage space. Repeated computation of these connections is out of the question, and storage on tape is also too slow to be practical. Even though the address of the S Unit has only six significant bits, it is unwieldy to store several addresses in the same word of core storage because of the computation necessary to select any one of them. Thus the only alternative was to reserve one word of core storage for each connection between S and A Units. Approximately 22,000 words were available for this function, and these were divided among 2760 A Units where each A Unit would have 8 input connections. Thus the parameters of the Perceptron are 64 Stimulus Units, 1 to 6 Response Units, and 2760 Association Units equally divided among the Response Units, with each A Unit having 8 connections to S Units.

4.3 The Flow Chart representing the Perceptron operation is shown in Figure 3. The first operation is an HPR to allow the sense switches to be set as desired for starting the program. This is followed by a test of sense switch 4. If this switch is down, binary tape 7 is read to insert a new set of S to A connections (essentially rewire) in the Perceptron. If the switch is up, binary tape 8 is read instead, which contains the last previously used set of S to A connections as well as the last previous A Unit Values and other Perceptron parameters. If tape 8 was read, sense switch 3 is tested next. If the switch is down a card (or a record on tape 0, depending on the setting of sense switch 5) is read containing 6 parameters defining the Perceptron operation. PLUS and MINUS control the steps taken in A Unit Values during the training process and may be used to modify the training scheme. THOLD is the fixed threshold level for all the A Units. INDEX4 contains R, the desired number of Response Units, and INDEX5 contains the number 2760/R, or the number of A Units connected to each R Unit. ZROVAL is a control word which is tested in the next step of the program. If ZROVAL is zero, all the A Unit Values in the machine are initialized at zero Value or at the Values read in from tape 8. If ZROVAL is not zero, initial Values are then read in from cards or tape 0. If sense switch 3 is up, the machine parameters and A Unit Values are left as they were read in from tape 8. The machine parameters are then printed out and the program stops to allow sense switch 5 to be reset or to allow tape 0 to be changed. The Perceptron is now initialized and ready to begin reading data.

4.4 The 64 S Unit excitations and the correct R Unit responses corresponding to one data point are then read in from a card or a record on tape 9. The first 64 card columns contain the S Unit inputs, then after two blank spaces columns 67-72 contain the R Unit correct answers. If only R of the six R Units are used, then the R correct answers should be put in columns 73-R to 72. Although the machine was primarily designed for binary inputs so the S Unit excitations would be 1's or 0's, it is possible to use any number from 0 through 9 for the excitation. Using multi-level data of this sort tends to emphasize the data points with the highest levels at any sample and ignore the lower level inputs. This may be desirable for some types of problems but not for others.

4.5 The next section of the program is the most critical, in that the computer spends practically all its time executing one small loop. The five-instruction loop is:

```
LDQ CONMAT+22080,1
STQ *+1
ADD **
TXI *+1,1,1
TIX *-4,4,1
```

Before each entry into the loop Index Register 4 is loaded with an 8 and the accumulator is cleared. The 22080 words of the CONMAT matrix contain +0400000004xx (or possibly +0402000004xx) where xx represents a random octal number corresponding to one of the S Units. The first two instructions in the loop move the word in CONMAT (under control of Index Register 1) into the next executable location taking it by way of the MQ. There the word is interpreted as an ADD (or SUB) instruction so that the contents of the xx S Unit are added to (or subtracted from) the accumulator. The next instruction increments IR #1, and the last tests IR #4 and transfers out of the loop after 8 iterations. The contents of the accumulator then equal the excitation to an A Unit. The threshold is subtracted and the net excitation is stored in an A Unit memory location. The A Units are counted by IR #2; and the program loops back, re-initializes IR #4 and the accumulator, and computes another A Unit excitation until all 2760 A Units have been handled. Thus the 5-instruction loop is iterated a total of 22080 times, and the remaining five instructions in the larger loop 2760 times, for each data input sample. The number of machine cycles for this part of the program is then $2[5(22080)+5(2760)]$ or 248,400 cycles. This requires almost 3 seconds of machine time, or considerably more than half the total 5.1 second computing time required for each data input.

4.6 Next the A Unit excitations are tested for sign, and their state (On if sign was plus, Off if minus) is stored. The Values of the On A Units are added, and the Values of the Off A Units are subtracted to compute the excitations to the R Units. Then the R Unit excitations are tested for sign, and the R Unit is put in its "1" state if the sign is plus and "0" if the excitation is negative. This completes computation of the Perceptron output decision; it is now necessary to check the decisions against the correct answers and to train the machine.

4.7 Each R Unit state is compared with the answer supplied on the data card, and a zero is placed in a checking storage to indicate a correct answer. If the answer is wrong, a "1" is stored and sense switch 1 is tested to determine whether the machine is to be corrected or not. If switch 1 is up, no adjustment is made to the Perceptron. If the switch is down, the

A Unit values are to be corrected in such a way as to increase the probability of a correct answer. The adjusting (or training) logic is as follows: If the output of an R Unit was zero when the desired output was a "1", then it is necessary to increase the excitation to the R Unit. This is done by adding a fixed step of magnitude PLUS to the Values of all the A Units which are "on" and which are connected to the given R Unit. A fixed step equal to MINUS is also added to the Values of the A Units which are "off" and are connected to the given R Unit. Thus the Values of the "on" A Units are increased, and the Values of the "off" A Units are decreased, both of which have the effect of increasing the excitation to the R Unit. If the R Unit output is "1" and the desired output is "0", then it is necessary to decrease the excitation to the R Unit. This is done by subtracting PLUS from the Values of the "on" A Units and subtracting MINUS from the "off" A Units. No changes are made in A Unit Values if the R Unit output agrees with the desired response. By adjusting the relative sizes of PLUS and MINUS, it is possible to achieve different sorts of reinforcement behavior. Notice that the training here is only in the direction of correcting the error and takes place in fixed steps. This differs from the procedure usually used at Cornell where the correction is made big enough to force the correct response before the machine goes on to the next data sample. In either case a number of runs through a given list of data points are usually necessary before perfect learning occurs, and the choice as to which of the techniques is preferable depends primarily on the application for which the Perceptron is intended.

4.8 After the Perceptron has responded to an input data sample and has been trained as necessary, a single line of output is written summarizing these operations. The first 64 entries in the line identify the input data point, and the following six bits correspond to the desired responses as read from the input card (or tape). The next group of six bits represents the R Unit states, and the last group of six bits indicates agreement (a zero) or disagreement (a one) of the R Unit states and the desired outputs. A single bit at the end of the line indicates the state of sense switch 1; being a "1" if the Perceptron was being trained or "0" if the machine was being tested without training. This output can be written only on tape 9 if sense switch 6 is up, or on both tape 9 and the on-line printer if the switch is down.

4.9 Finally sense switch 2 is tested as a method of ending the computer run. If switch 2 is up, the computer reads another input card (or tape record) and repeats its operations on the next data sample. When switch 2 is put in its down position, the computer will stop reading input data and transfer to an unloading routine. This routine prints out the Values of all the

A Units on the output tape (tape 9) and/or the printer; then it stores the S to A connection matrix (CONMAT), the A Unit Values, and the other Perceptron parameters on binary tape 8 to allow temporary suspension of running while maintaining stored Values and other parameters. Notice that the A Units are stored backwards in the machine. Thus the A Unit in the lowest machine address (and whose Value is first in the printout) is connected to the last R Unit (whose outputs appear in the sixth bit of the R Unit group in the printout), and its inputs are represented by the last eight entries in the CONMAT list. Before returning to the BELL System, the machine stops once more to allow the sense switches to be returned to their normal positions. Then an automatic return to BELL System control allows sequencing of the next job into the computer.

GENERATION OF RANDOM PERCEPTRON CONNECTIONS

5.1 Since the Perceptron program requires as one of its inputs a list of the S to A cross-connections (CONMAT), a program was written to generate this connection matrix. The FORTRAN 3 program shown in Appendix 2 generates this list from a random number subroutine and writes the CONMAT list on binary tape 7. The connections to each A Unit are independent of those to other A Units, but the eight inputs to each A Unit are selected such that no two of them connect to the same S Unit. The particular random set of connections chosen depends on the octal number Q which is fed in to start the routine.

5.2 When the octal number INST on the input card represents ADD 400, the word stored on the binary tape corresponding to each entry in the CONMAT list is +0400000004xx, where the xx represents the octal number (0 to 77) identifying the S Unit involved in the connection. The Perceptron program translates this by adding the output of S Unit xx (octal) to the excitation of the proper A Unit. This simulates the operation of a Perceptron using only the uncomplemented outputs of the S Units. It is possible to modify the CONMAT program slightly to simulate both the S Unit outputs and their complements by mixing both ADD and SUB instructions in the CONMAT tape. Thus the words on the binary tape would be either of the form +0400000004xx or alternately +0402000004xx, the form used for each connection to be selected systematically or at random.

5.3 A printout list of the connections is also provided for reference, where a list of eight decimal numbers (0 to 63) is given for each A Unit, identifying the S Units to which it is connected. This list is in the same order as the binary list, and so the first group of eight numbers in the CONMAT listing corresponds to the last A Unit in the Perceptron program. The first eight octal instruction words are also printed on line as a checking feature on the program.

COMPUTER SIMULATION OF THE HALF-PERCEPTRON

6.1 The half-Perceptron program shown in Appendix 3 is designed to have inputs and outputs compatible with the Perceptron program. It too has 64 S Units and 6 R Units, but its logical behavior is governed by the coefficients stored in its 64x6 element S to R matrix. Because of its simpler logical structure, the half-Perceptron runs at the rate of approximately 0.5 second per data sample as opposed to the 5.1 seconds per sample for the Perceptron. It also becomes practical to print out the internal structure (MATRIX) of the half-Perceptron periodically during training.

6.2 Figure 4 shows the flow chart of the half-Perceptron program. As in the Perceptron, the first instruction that is encountered after control is transferred from the BELL system is a STOP to allow the sense switches to be set for the program. If sense switch 4 is up, the MATRIX is read in from binary tape 8, which may contain the MATRIX elements from a preceding run. If the switch is down, sense switch 3 is then tested to initialize the MATRIX. If switch 3 is up, all MATRIX elements are assumed to be zero; if the switch is down, the MATRIX is read into the machine from the card reader. The half-Perceptron is now initialized and ready to read input data.

6.3 The input data sample and the desired responses are then read from a card (if switch 5 is down) or from tape 0 (if switch 5 is up) in the same format as that used in the Perceptron. The only significant difference is that the half-Perceptron always assumes that six responses are required, and blank spaces on the ANSWER field of the card will be interpreted as required zeros. The only penalty paid for the assumed six R Units in the half-Perceptron is a small time loss. In the Perceptron it was definitely advantageous to divide up the available A Units among only the required number of R Units.

6.4 The R Unit states are determined by multiplying each S Unit excitation by the appropriate MATRIX element and determining whether the net excitation to the R Unit is positive or negative. The checking and training sections are essentially the same as those in the Perceptron. Each R Unit state is compared with the desired response, and the results of the comparison stored in the CHECK locations. If an R Unit state is wrong, sense switch 1 is tested to determine whether the MATRIX elements should be adjusted. If the switch is down, the training procedure is as follows. If the R Unit state is zero and the desired response is a "1", then the excitation to the R Unit should be increased. Thus the MATRIX elements associated with that R Unit are each increased by the amount of excitation to the associated S Unit. The reason for this is that S Units which have zero excitation are not affecting the R Unit excitation, and there is no justification for increasing the strength of their connections to the R Unit. If the S Unit excitations are allowed to range from 0 through 9 (rather than just being binary inputs), then the S Units having the largest excitation will have their connections strengthened the most. There is some justification for this, although the training procedure is designed primarily for binary inputs. If the R Unit state is a "1" where a "0" is desired, the excitation should be decreased. Thus each MATRIX element is decreased by the amount of the excitation to the associated S Unit. If the R Unit state agrees with the desired response, no change is made in the MATRIX elements.

6.5 The printout after each data sample is processed is identical to that of the Perceptron, containing the 64 S Unit excitations, the 6 desired responses, the 6 R Unit states, the 6 checking results, and the single bit indicating whether the half-Perceptron was being trained or merely tested. Output is on tape 9, and may also be printed on line by placing sense switch 6 down. After each data sample is processed and the results printed, sense switch 2 is tested to determine whether the next data sample should be read (up) or the running terminated (down). If the switch is down the MATRIX elements are printed out on tape 9 (and on line if switch 6 is down), and are also listed in binary on tape 8 so they can be stored for later resumption of running. The program then stops to allow the sense switches to be reset, and then transfers back to the BELL System. During the running of the half-Perceptron, the MATRIX is also printed out on tape 9 (and optionally on line) after each 100 data samples have been processed. After this printout the machine continues running. The MATRIX list is always used and printed in the same order, so the first 64 entries in the list correspond to the connections from the S Units (in ascending numerical order) to the first R Unit.

EXPERIMENTAL EVALUATION OF THE PERCEPTRON AND HALF-PERCEPTRON

7.1 An experimental problem was desired to test the capabilities of the Perceptron and the half-Perceptron. The simplest logical problem that can be given a classification machine is to require that each of its outputs be identical to one of its inputs, and thus independent of all other inputs. This was the test problem used for the two learning machines, and the states of the six R Units were required to be identical to the states of the first six S Units. The FORTRAN program in Appendix 4 generates the input data for the problem. A random list of 64 binary numbers is generated from the random number routine and used as the S Unit excitation. The first six of these are also used as the desired responses, and the two groups are written on tape 6 in the proper format for input to the Perceptron or half-Perceptron programs. A list of 1000 data samples was prepared in this way.

7.2 A measure of the degree of learning is given by the number of errors occurring in a fixed number of data samples. The errors were counted in groups of ten input samples, and the number of errors in each group is plotted as a function of the total number of input samples in Figure 5 for the half-Perceptron and Figure 6 for the Perceptron. In the half-Perceptron case all 1000 input samples were processed in direct succession. In the Perceptron the first 237 samples were processed in one run, then in the second run the first 520 samples were processed. Thus between input samples 238 and 474 the machine was processing input data that it had already seen once before, while other data samples were only being processed for the first time. The fact that there is no noticeable discontinuity in the learning curve either at the 237 or the 474 sample points verifies the fact that the learning process chosen is only an incremental training and that more than one pass through a given set of data is necessary before it is fully assimilated.

7.3 The learning curves for the half-Perceptron and the Perceptron are similar in general form, beginning from a point in the vicinity of the 30 errors (in sixty possible bits) that would be predicted from purely random operation and decaying more or less exponentially toward some lower error rate. The initial slopes for the two machines are roughly the same, although the half-Perceptron reaches a higher degree of learning much sooner than the Perceptron. Notice that in the half-Perceptron the error rate gets quite small, and that the machine actually made no errors at all in the last 100 samples. The Perceptron, on the other hand, only approaches an error rate of about 4 errors per 60 bits and seems to maintain this level of learning.

7.4 The difference in final learning ability of the two machines is due to the difference in logical organization of the machines, and to the special nature of this particular problem. What is really desired in this problem is that the connections in the logical network from S Unit X to R Unit X be strengthened, and that the connections from all the other S Units to that R Unit be zero or canceled. Because of the organization of the half-Perceptron, this can actually be done in this machine; and in fact it can be seen directly from the MATRIX elements. In the Perceptron, however, each path from a given S Unit to a particular R Unit must pass through an A Unit to which other S Units are connected. Thus, while the values of all A Units which do not provide transmission paths between the desired S and R Units can be made zero, it is impossible to eliminate the noise produced by the other S Unit excitations as they pass through A Units which do provide transmission paths for the desired S Unit. This noise can only be reduced by the statistical averaging from a large number of A Units and goes down as the square root of the number of A Units. Thus the half-Perceptron can have a zero error rate for this simple class of problem but the Perceptron organization limits the machine to some non-zero error rate determined by the number of A Units in the machine.

7.5 Another significant difference between the two machines is the difficulty in interpreting the information stored in the Values of the Perceptron or in the MATRIX of the half-Perceptron. In the sample problem where six of the 384 connections in the half-Perceptron were to be strong relative to the others, this result shows up very clearly in the MATRIX printout. After the 1000 training samples the six desired connections had coefficients ranging between 31 and 38, while the remaining 378 elements of the MATRIX had an average value of -0.541 and a standard deviation around this average of 1.43. The negative average value of the weak connections is due to the tendency for the average of the MATRIX elements to be zero. In the Perceptron the connections between the desired S and R Units (and thus the learning) are distributed throughout the machine. In this problem the learning can be demonstrated by dividing the A Units belonging to a given R Unit into two classes; those which connect to the desired S Unit, and those which do not. This was done for the A Units connected to R Unit #5 and the distributions of the Values plotted in Figure 7 for each of the two classes. Neither distribution differs significantly from the normal distribution. The mean of the Values of those A Units (393) which do not connect to S Unit #5 is -1.8 and the standard deviation of the Values is 4.7. The mean for the 65 units which did connect to S Unit #5 was 14.3 and the standard deviation was 5.0. The two standard deviations are not significantly different, but the different means show that a significant difference has appeared between the two classes of A Units, indicating that learning has taken place. The standard deviations are also significantly smaller than $\sqrt{76}$ or 8.72, which would be predicted by random walk considerations, indicating that the Values are converging toward some fixed distribution and will not continue to vary indefinitely. In more complicated problems the learning will be even more deeply embedded in the A Unit Values, and in practical cases it may be impossible to recognize the various classes of A Units from the distribution of their Values.

7.6 The sample problem just discussed is a very simple one, and is rather easily solved by both the Perceptron and the half-Perceptron. A second problem was designed in order to demonstrate the limits of capability of the two machines. In this problem only two R Units were used, and in the Perceptron the A Units were divided between these, giving 1380 A Units to each R Unit. The desired outputs of the machines were defined as follows: If more of the first five S Units were in the one state than the number of the next five S Units which were in the one state, the output of R Unit #5 should be a one. Otherwise (more ones in the second group of five than in the first group of five, or an equal number in each) R Unit #5 should be zero. R Unit #6 was required to be "1" if the number of ones in the first six S Units was odd and zero if the number of ones was even. The requirement on R Unit #5 is still among the class of problems which may be solved by a half-Perceptron, and is also within the capabilities of the Perceptron, where again the Perceptron is likely to exhibit more random noise in its output because of its organization. The required solution for R Unit #6, however, is one of the most difficult switching functions of six variables; and it may be shown that it is beyond the theoretical capability of the half-Perceptron. While a sufficiently large and general Perceptron can handle this problem, it becomes very difficult for a Perceptron of the size and limited organization used here.

7.7 A tape was made from the input tape for the previous problem, substituting new required R Unit responses according to the above definitions. The FORTRAN 3 program used to produce this tape is shown in Appendix 5. This tape was processed by both the half-Perceptron and the Perceptron, and the learning curves are shown in Figures 8 and 9 respectively. The errors on each of the two R Units are plotted separately, and the vertical axis on the plots is the number of errors on the given R Unit in each group of 20 samples. The upper curve in each Figure represents the errors in R Unit #5 and the lower is for R Unit #6. The error rate for R Unit #5 in the half-Perceptron is around 2 or 3 errors per 20 samples. This of course indicates that the problem is being solved by the machine, although convergence is not as rapid as in the earlier problem of Figure 5. The errors per 20 samples in R Unit #6 continue to fluctuate around 10, which indicates that the machine is merely guessing at random and that no solution is being formed.

7.8 Only the first 400 samples of the tape were processed on the Perceptron, and the errors are plotted in Figure 9. Again the errors in R Unit #5 are decreasing, and the error rate after 400 samples is about 5 per 20 samples and not significantly different from the error rate in the half-Perceptron after 400 samples. Also the plot of R Unit #6 errors for the Perceptron shows no indication of learning after 400 samples. It is not clear whether this rate would decrease with a much larger number of input samples, but for this size Perceptron it is unlikely that the final error rate after a very large number of samples would get very much below 10. It should be observed that the relative number of required zeros to ones for R Unit #5 is approximately 5 to 3 because of the zero requirement when both groups of S Units have an equal number of ones. Thus a learning machine could have an error rate as low as 7.5 per 20 samples merely by observing this bias and guessing all zeros. However, there is no evidence from the data that either machine tried this tactic.

CONCLUSIONS

8.1 The sample problems shown here demonstrate the type of performance to be expected from learning machines when they are given simple problems as in the first example, or when they are given problems requiring or exceeding their full capabilities. The fact that the half-Perceptron reached a lower error rate on the first example than the Perceptron did, and that the performance was similar to that of the Perceptron in the second example, should not be considered an advantage of the half-Perceptron type of construction. The reason is that the half-Perceptron is capable of solving only a limited class of problems, and that its more direct organization allows a lower error rate for the problems for which it is designed. The majority of practical classification problems, particularly problems which require recognizing switching functions, are likely to exceed the capabilities of the half-Perceptron; and a learning machine having the flexibility of the Perceptron would be required.

8.2 It is possible that a machine concept could be developed having the flexibility of the Perceptron organization, yet avoiding some of the difficulties inherent in its organization. However these difficulties tend to disappear as the number of A Units becomes very large due to averaging out of the random noise components in the A Units. It is presently impractical to consider a much larger number of A Units in a computer simulation because of speed and storage limitations, and present electromechanical realizations of A Unit operation are too bulky and expensive to allow very large Perceptrons. However, it is reasonable to expect that some sort of solid state device could be developed having the logical and memory characteristics of an A Unit. If such a device could be built along the general lines of multi-hole magnetic cores and/or thin film semi-conductor circuitry, Perceptrons could be considered having 10^6 or more A Units. In such machines very powerful logical behavior and learning capability would be available for solving various kinds of classification problems.

8.3 The use of machines such as the Perceptron for naval problems such as target classification would be restricted to laboratory studies. It is not practical to use learning machines as such on board operational vessels because the amount of information available (in the form of signals from identifiable targets) to each vessel would not be sufficient to train such a machine. Rather the proper use of the learning machines would be to pool the information available from all sources (tape recordings from ships or shore-based installations) and to feed all this information to a single learning machine for comparative analysis. In the research phase of target classification studies, the learning machines could be used to determine which characteristics of the input data were significant in classifying targets. In later operational use of such equipment, the laboratory machine would constantly modify its structure in such a way as to maintain an optimum classification scheme, based on the currently available input data. Measurements made on the laboratory machine would then provide information for designing, or perhaps simply readjusting, non-adaptive operational classifiers for the ships of the fleet.

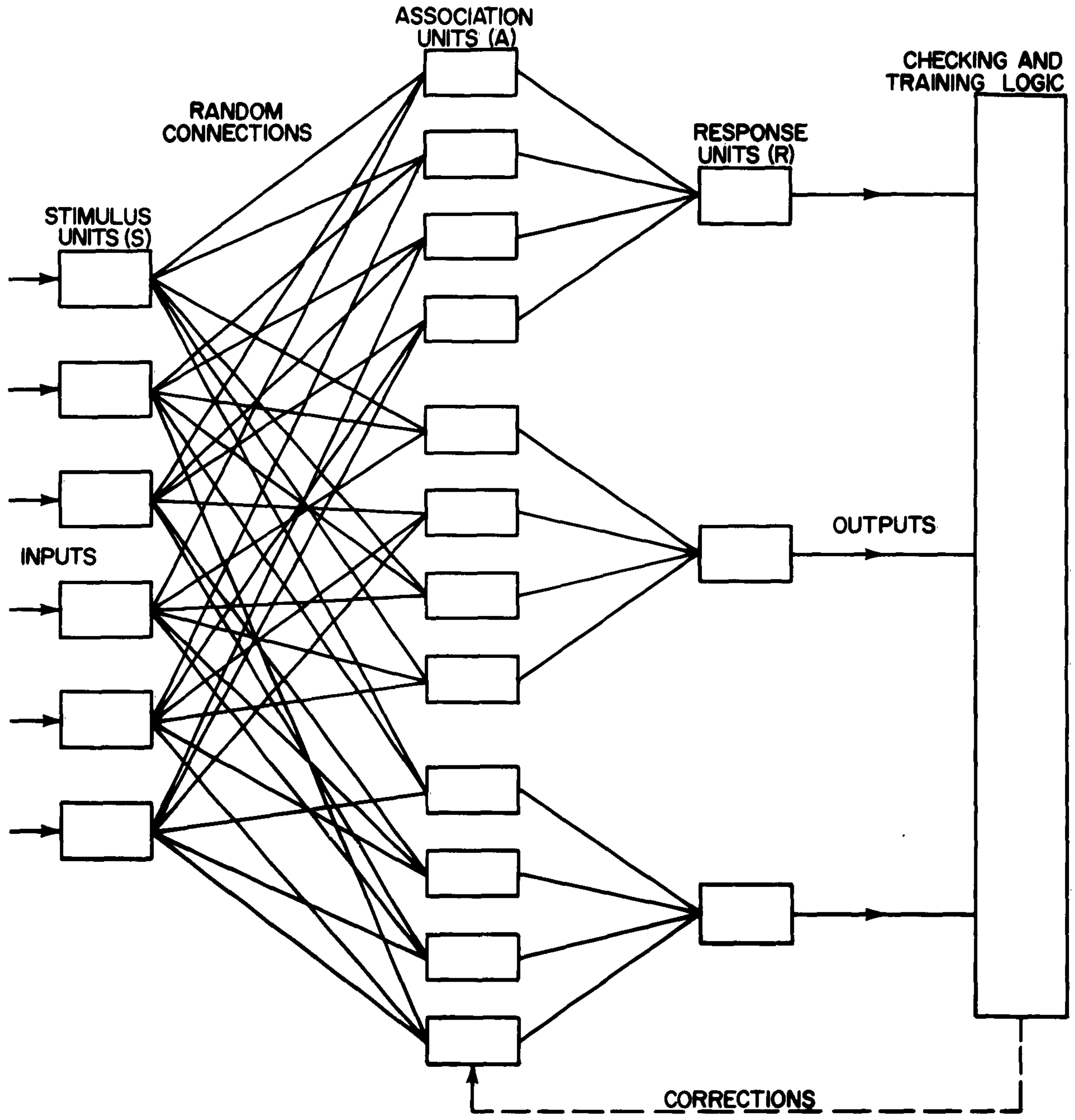


FIG. 1 PERCEPTRON BLOCK DIAGRAM

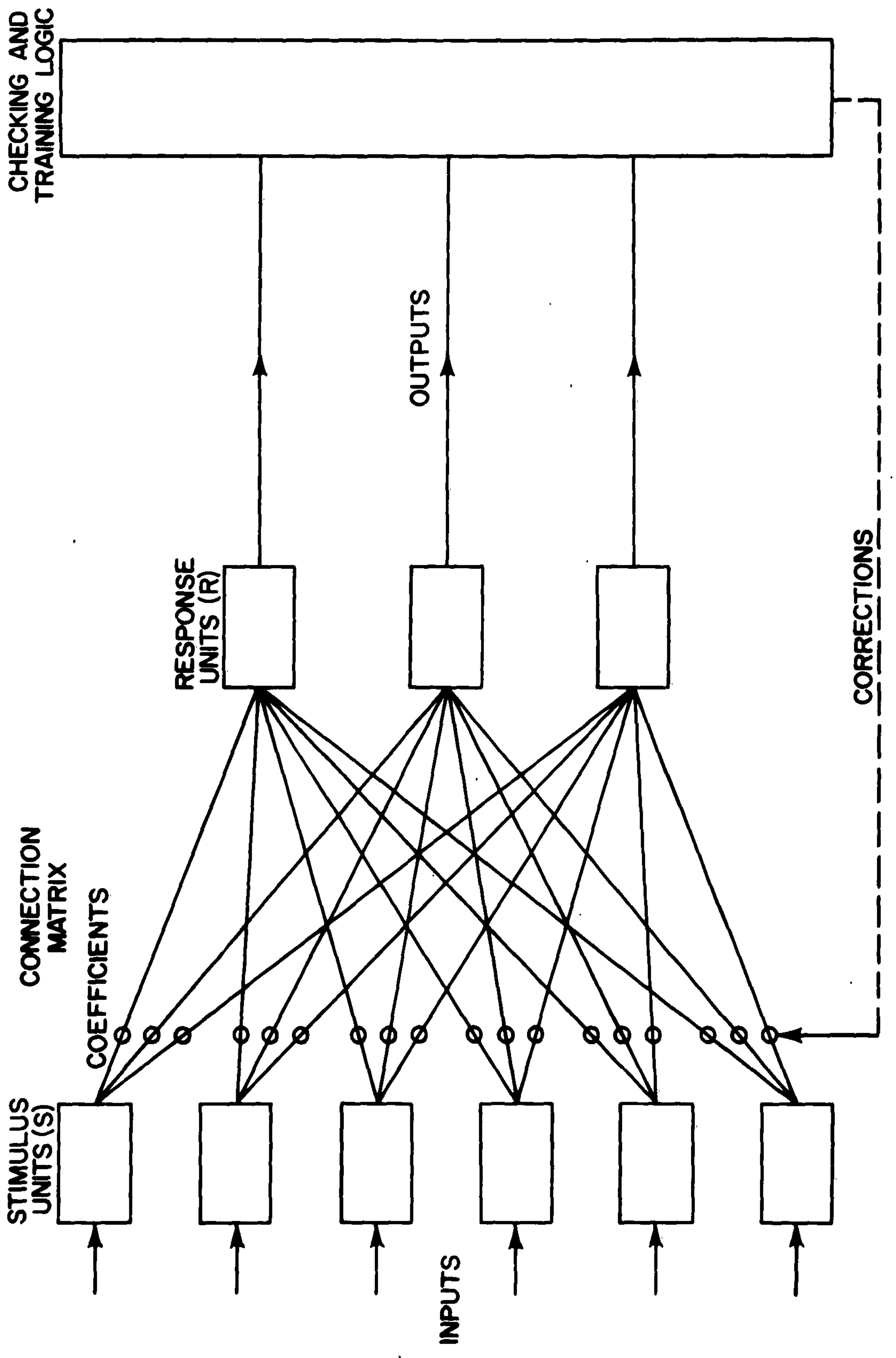
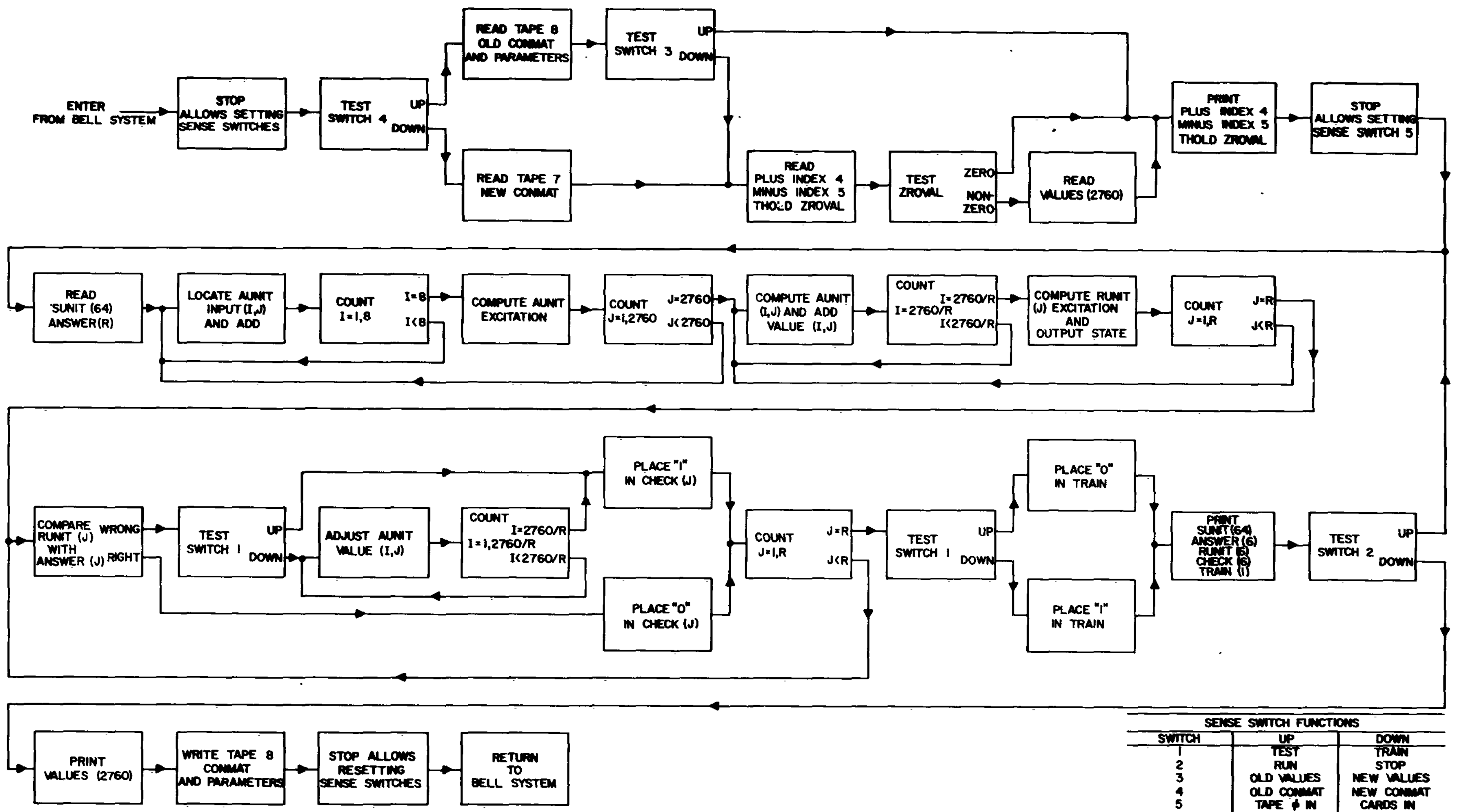


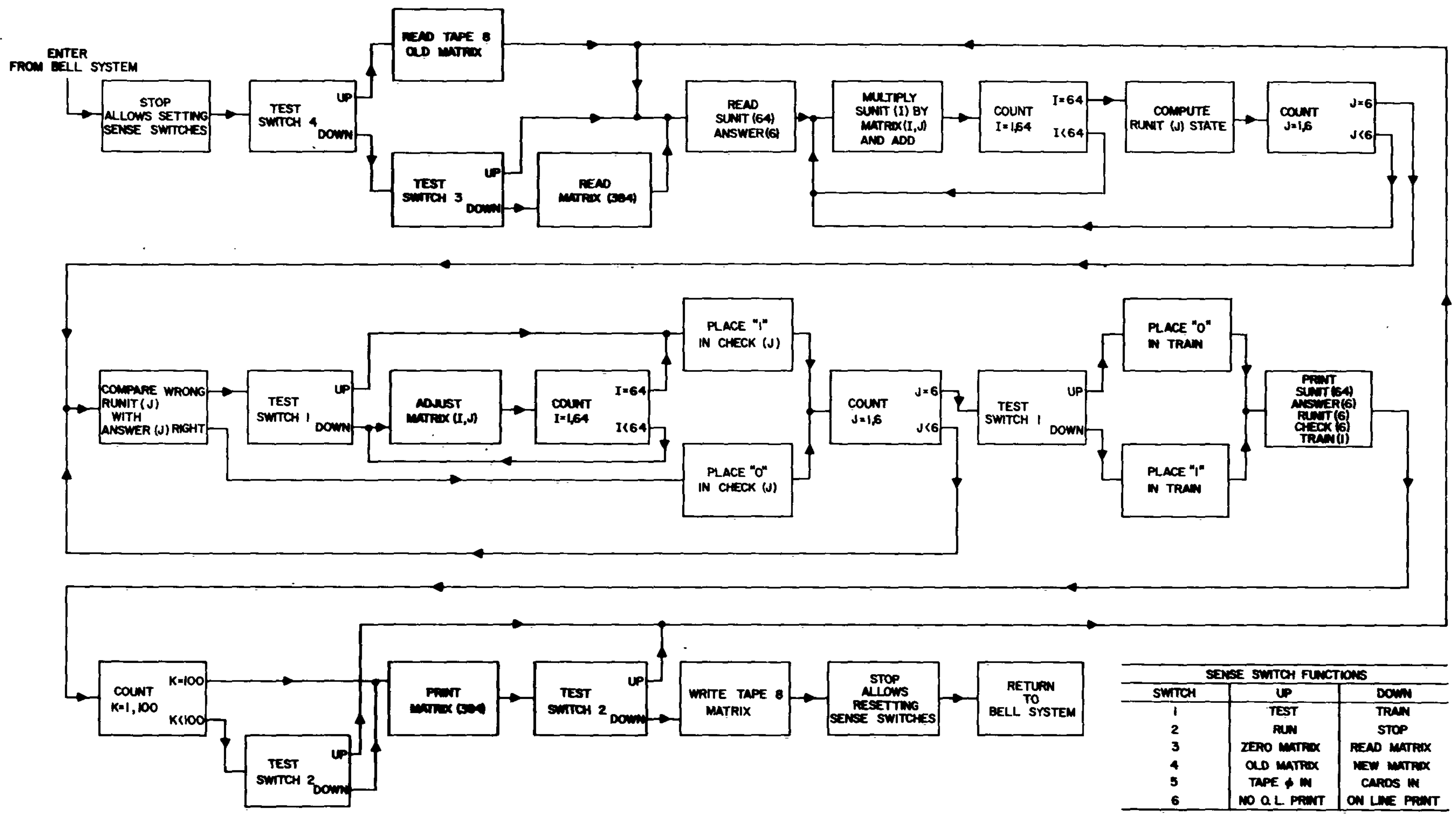
FIG. 2 HALF-PERCEPTRON BLOCK DIAGRAM


ENTER FROM BELL SYSTEM
STOP ALLOWS SETTING SENSE SWITCHES
TEST SWITCH 4 UP DOWN
READ TAPE 8 OLD CONMAT AND PARAMETERS
READ TAPE 7 NEW CONMAT
TEST SWITCH 3 UP DOWN
READ PLUS INDEX 4 MINUS INDEX 5 THOLD ZROVAL
TEST ZROVAL ZERO NON-ZERO
READ VALUES (2760)
PRINT PLUS INDEX 4 MINUS INDEX 5 THOLD ZROVAL
STOP ALLOWS SETTING SENSE SWITCH 5
READ SUNIT (64) ANSWER(R)
LOCATE AUNIT INPUT (I,J) AND ADD
COUNT I=1,8 I=8 I<8
COMPUTE AUNIT EXCITATION
COUNT J=1,2760 J=2760 J<2760
COMPUTE AUNIT (I,J) AND ADD VALUE (I,J)
COUNT I=2760/R I=2760/R I<2760/R
COMPUTE RUNIT (J) EXCITATION AND OUTPUT STATE
COUNT J=1,R J=R J<R
COMPARE RUNIT (J) WITH ANSWER (J) WRONG RIGHT
TEST SWITCH 1 UP DOWN
ADJUST AUNIT VALUE (I,J)
COUNT I=1,2760/R I=2760/R I<2760/R
PLACE "1" IN CHECK (J)
PLACE "0" IN CHECK (J)
COUNT J=1,R J=R J<R
TEST SWITCH 1 UP DOWN
PLACE "0" IN TRAIN
PLACE "1" IN TRAIN
PRINT SUNIT(64) ANSWER(6) RUNIT (6) CHECK (6) TRAIN (1)
TEST SWITCH 2 UP DOWN
PRINT VALUES (2760)
WRITE TAPE 8 CONMAT AND PARAMETERS
STOP ALLOWS RESETTING SENSE SWITCHES
RETURN TO BELL SYSTEM

SENSE SWITCH FUNCTIONS

SWITCH	UP	DOWN
1	TEST	TRAIN
2	RUN	STOP
3	OLD VALUES	NEW VALUES
4	OLD CONMAT	NEW CONMAT
5	TAPE ϕ IN	CARDS IN
6	NO O.L. PRINT	ON LINE PRINT

FIG. 3 PERCEPTRON FLOW CHART



SENSE SWITCH FUNCTIONS		
SWITCH	UP	DOWN
1	TEST	TRAIN
2	RUN	STOP
3	ZERO MATRIX	READ MATRIX
4	OLD MATRIX	NEW MATRIX
5	TAPE ϕ IN	CARDS IN
6	NO O. L. PRINT	ON LINE PRINT

FIG. 4 HALF-PERCEPTRON FLOW CHART

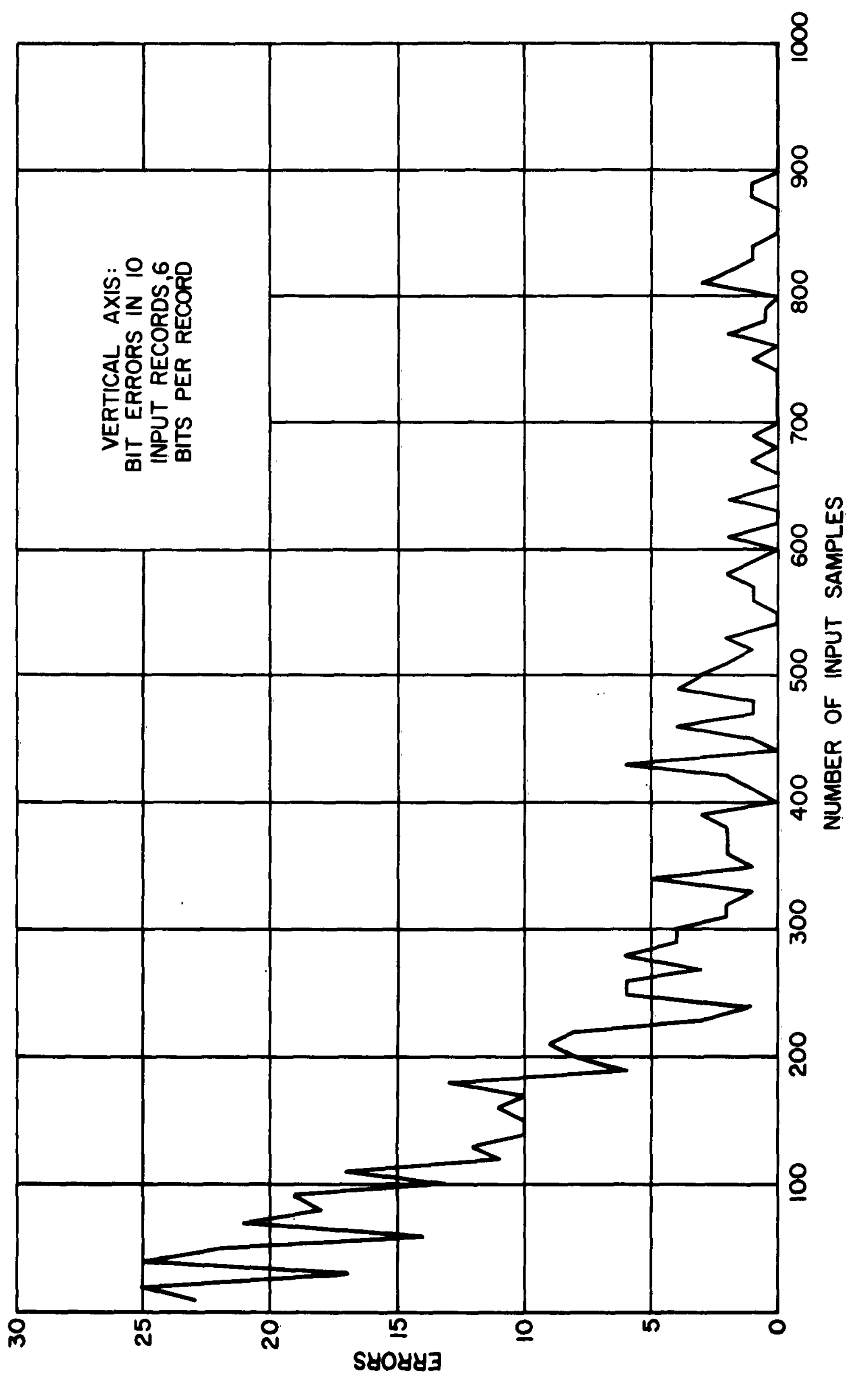


FIG. 5 HALF PERCEPTRON LEARNING CURVE : ERROR RATE VERSUS NUMBER OF INPUT RECORDS, PROBLEM I

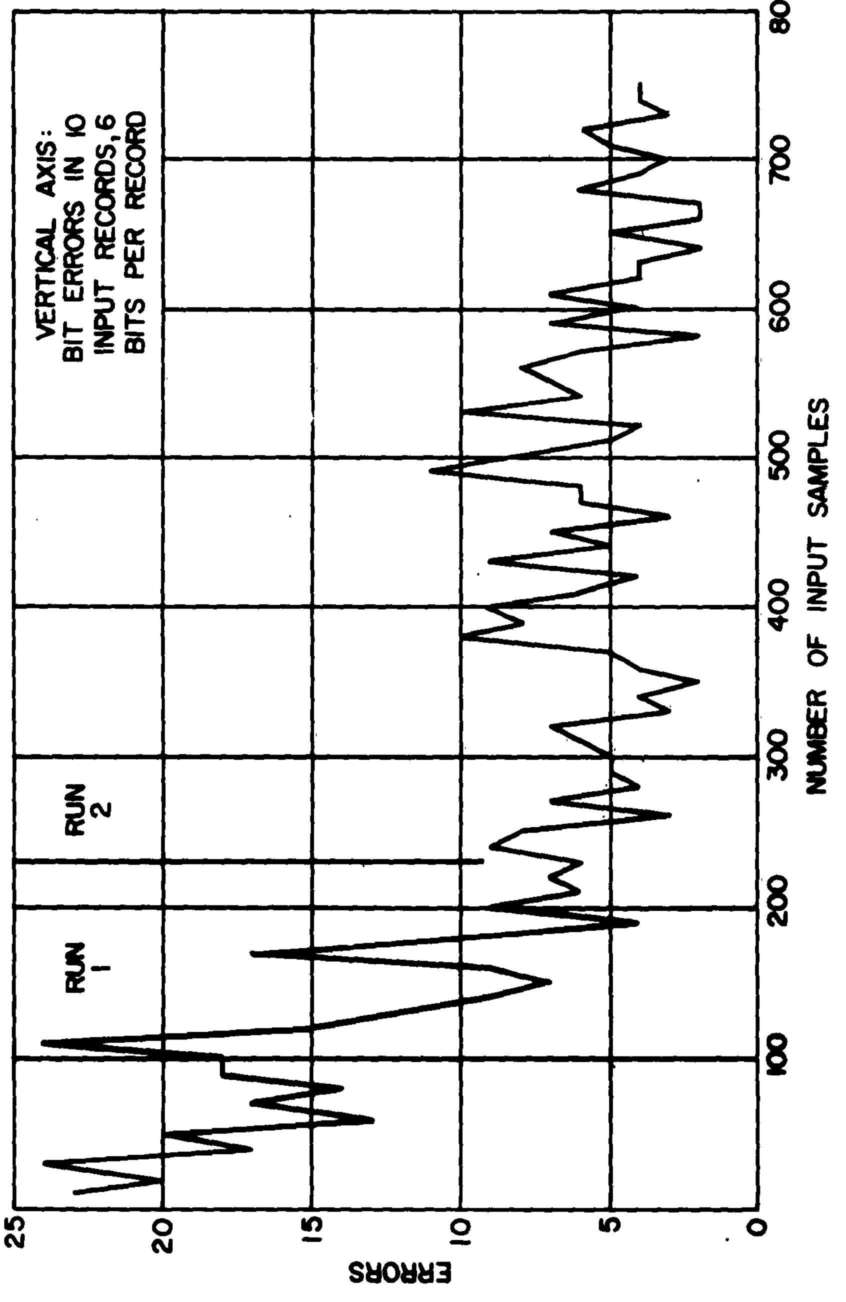


FIG. 6 PERCEPTRON LEARNING CURVE: ERROR RATE VERSUS NUMBER OF INPUT RECORDS, PROBLEM I

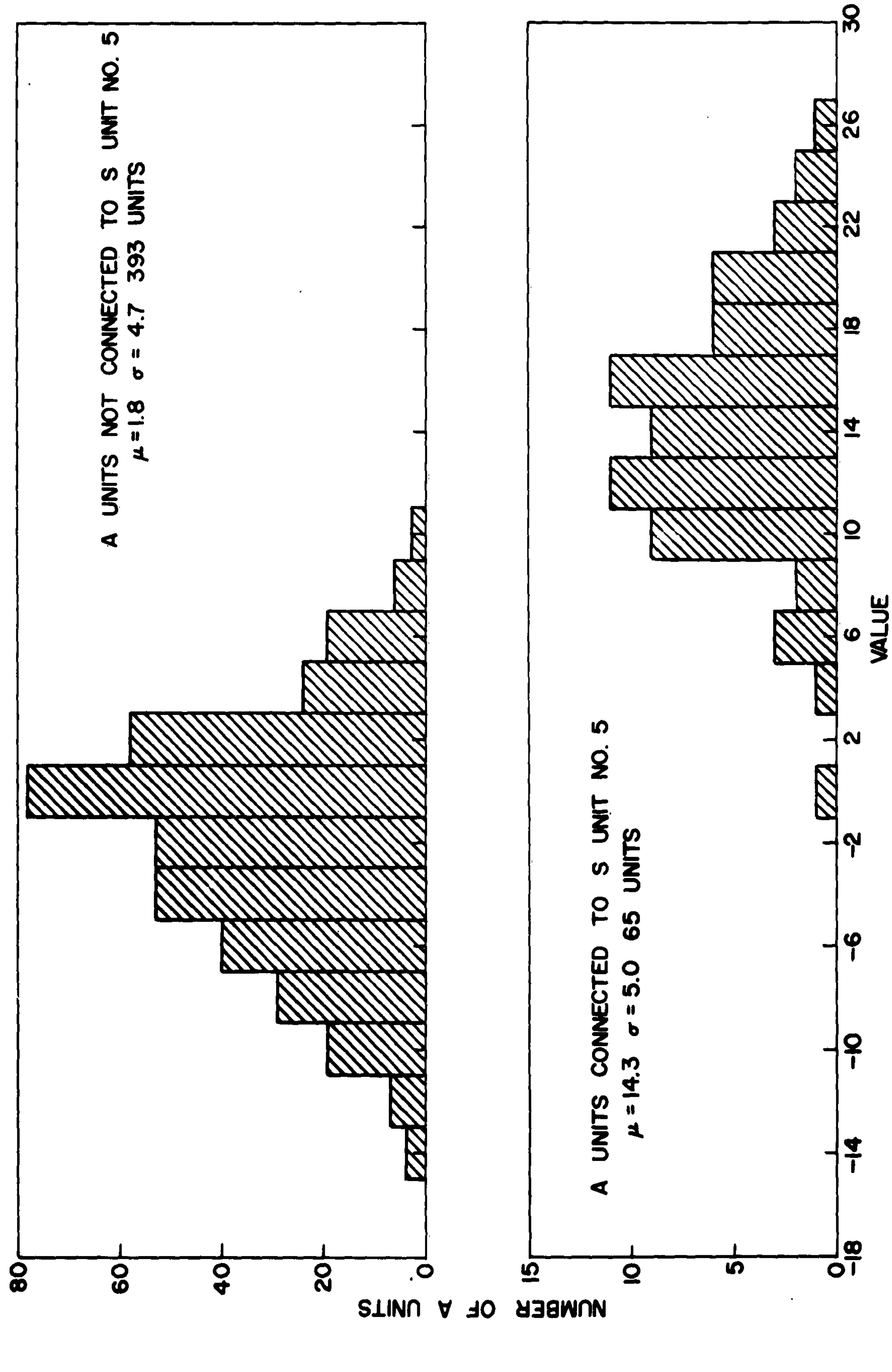


FIG. 7 DISTRIBUTION OF A UNIT VALUES: A UNITS CONNECTED TO R UNIT NO. 5, 76 WRONG ANSWERS CORRECTED

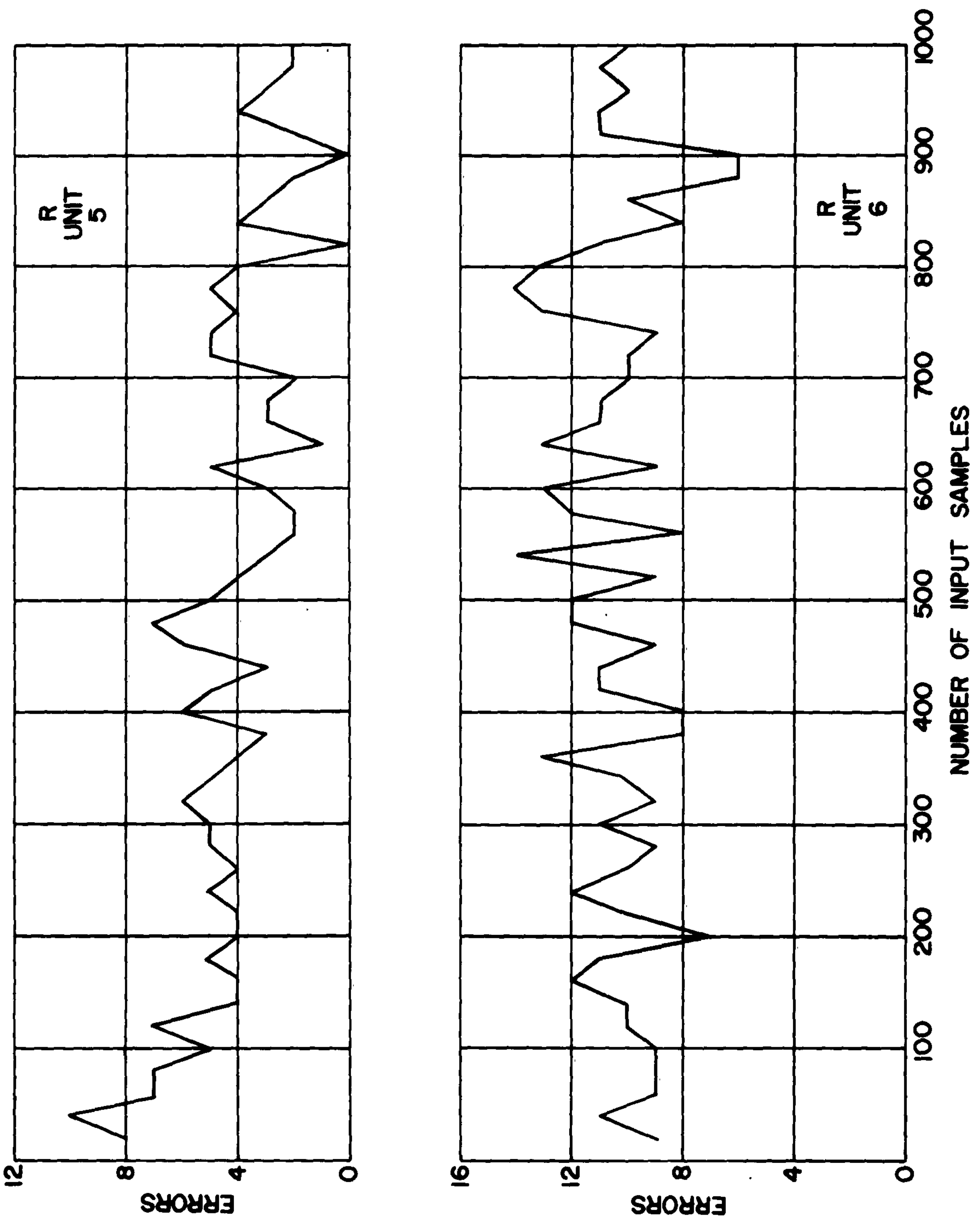


FIG. 8 HALF-PERCEPTRON LEARNING CURVE: ERRORS PER 20 INPUT SAMPLES, PROBLEM 2

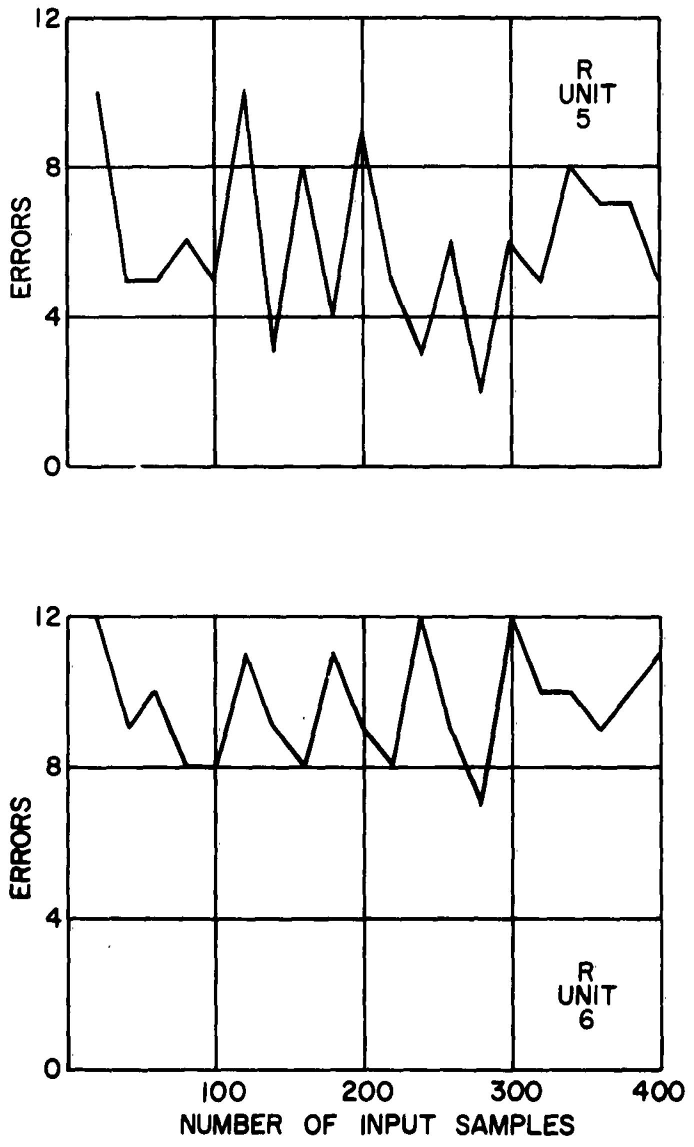


FIG. 9 PERCEPTRON LEARNING CURVE: ERRORS PER 20 INPUT SAMPLES, PROBLEM 2

APPENDIX 1   PERCEPTRON MAIN PROGRAM

```
        ORG 24
        HPR
        SWT 4                         TEST FOR REBUILD
        TRA RETAPE
        REM READ NEW CONNECTION MATRIX
        LXA INDEX2,1
        RTB 7
        CPY CONMAT+22080,1
        TIX *-1,1,1
        TRA NEWVAL
        REM READ OLD PARAMETERS AND CONNECTION MATRIX
RETAPE  LXA INDEX3,1
        RTB 8
        CPY VALUE+24846,1
        TIX *-1,1,1
        NOP
        SWT 3                         TEST FOR RESET VALUES
        TRA FIRST
        REM READ NEW SYSTEM PARAMETERS
NEWVAL  TSX XINPUT,4
        NTR F4,,0
        NTR PLUS
        NTR MINUS
        NTR THOLD
        NTR INDEX4
        NTR INDEX5
        NTR ZROVAL
        XIT XXXRET
        CLA ZROVAL
        TZE *+4
        TSX XINPUT,4
        NTR F2,,0
        MON VALUE,,VALUE+2759
FIRST   TSX OUTPUT,4                  PRINT PERCEPTRON CONSTANTS
        NTR F4,,0
        NTR PLUS
        NTR MINUS
        NTR THOLD
        NTR INDEX4
        NTR INDEX5
        NTR ZROVAL
        XIT XXXRET
        LXD INDEX5,1
        SXD RIGHT-2,1
        SXD RIGHT+1,1
        HPR
        REM READ NEXT STIMULUS AND COMPUTE A INPUTS
 STIM   TSX XINPUT,4
        NTR F3,,0
        MON SUNIT,,ANSWER+5
        LXD ONE,1
        LXD INDEX2,2                  LOAD 2760
```

```
NEXTA   LXD INDEX3,4                      LOAD 8
        CLA ZERO
        LDQ CONMAT+22080,1
        STQ *+1
        ADD                               ADDRESS COMPUTED BY PROGRAM
        TXI *+1,1,1                       INCREMENT CONNECTION
        TIX *-4,4,1                       COUNT INNER LOOP
        SUB THOLD
        STO AUNIT+2760,2
        TIX NEXTA,2,1                     COUNT A UNITS
        REM COMPUTE R UNIT INPUTS AND OUTPUTS
        LXD ONE,1
        LXD INDEX4,2                      LOAD R
NEXTR   LXD INDEX5,4                      LOAD 2760/R
        CLA ZERO
TESTA   LDQ AUNIT+2760,1
        TQP AON
        SUB VALUE+2760,1
        LDQ MINUS
        STQ AUNIT+2760,1
        TXI *+1,1,1                       COUNT A UNITS
        TIX TESTA,4,1
        TRA STORER
AON     ADD VALUE+2760,1
        LDQ PLUS
        STQ AUNIT+2760,1
        TXI *+1,1,1                       COUNT A UNITS
        TIX TESTA,4,1
STORER  TPL RON
        CLA ZERO
        TRA *+2
RON     CLA ONE
        STO RUNIT+6,2
        TIX NEXTR,2,1
        REM CHECK ANSWERS AND TRAIN
        LXD INDEX4,1                      LOAD R
        LXD INDEX5,2                      LOAD 2760/R
TESTR   CLA RUNIT+6,1
        SUB ANSWER+6,1
        TZE RIGHT
        SWT 1                             TEST FOR TRAINING
        TRA RIGHT-1
        LXD INDEX5,4
        TMI INCRE
        CLS AUNIT+2760,2
        ADD VALUE+2760,2
        STO VALUE+2760,2
        TXI *+1,2,-1
        TIX *-4,4,1                       COUNT A UNITS FOR GIVEN R UNIT
        TRA RIGHT-2
INCRE   CLA AUNIT+2760,2
        ADD VALUE+2760,2
        STO VALUE+2760,2
        TXI *+1,2,-1
        TIX *-4,4,1                         COUNT A UNITS FOR GIVEN R UNIT
        TXI *+1,2,
        CLA ONE
RIGHT   STO CHECK+6,1
        TXI *+1,2,
        TIX TESTR,1,1                       COUNT R UNITS
        CLA ONE
        SWT 1                               TEST FOR TRAINING
        CLA ZERO
        STO TRAIN
        TSX OUTPUT,4
        NTR F1,,0
        MON SUNIT,,TRAIN
        SWT 2                               TEST FOR STOP
        TRA STIM
        TSX OUTPUT,4                        PRINT PRESENT VALUES
        NTR F2,,0
        MON VALUE,,VALUE+2759
        REM WRITE PARAMETERS AND CONNECTION MATRIX ON TAPE 8
        LXA INDEX3,1
        WTB 8
        CPY VALUE+24846,1
        TIX *-1,1,1
        HPR
        TSX RETURN,4
        HTR *-1
ZERO    OCT 000000000000
ONE     OCT 000001000000
INDEX2  OCT 005310053100                    2760  22080
INDEX3  OCT 000010060416                    8  24846
F1      BCD 5(10X,64I1,3(3X,6I1),3X,I1)
        OCT 777777777777
F2      BCD 1(12I6)
F3      BCD 3(64I1,2X,6I1)
        OCT 777777777777
F4      BCD 1(6I6)
        ORG 256                             OCTAL 400
SUNIT   BSS 64
ANSWER  BSS 6
RUNIT   BSS 6
CHECK   BSS 6
TRAIN
AUNIT   BSS 2760
VALUE   BSS 2760
CONMAT  BSS 22080
INDEX4                                      NUMBER OF R UNITS (R)
INDEX5                                      2760/R
PLUS
MINUS
THOLD
ZROVAL
        END 24
```

APPENDIX 2 PERCEPTRON CONNECTION MATRIX PROGRAM

```
      DIMENSION LIST(8,2760)
      READ 100,Q,INST
      NO=0
      DO 40 J=1,2760
      DO 40 I=1,8
      N=I-1
   10 CALL NUMBER(Q,R)
      IR=R*64.
      IF(N)70,40,20
   20 DO 30 K=1,N
      IF(IR-LIST(K,J))30,10,30
   30 CONTINUE
   40 LIST(I,J)=IR
      PRINT 110,((LIST(I,J),I=1,8),J=1,2760)
S     LXD NO,(I)
S     WTB 7
S  50 CLA LIST,(I)
S     ARS 18
S     ADD INST
S     STO LIST,(I)
S     CPY LIST,(I)
S     TXI*60,(I),1
S  60 TXL*50,(I),22079
S     WEF 7
      WRITE OUTPUT TAPE 11,120,(LIST(I,1),I=1,8)
      CALL RETURN
   70 STOP
  100 FORMAT(2012)
  110 FORMAT(4(4X,8I3))
  120 FORMAT(10X,8012)
        END
```

APPENDIX 3   HALF-PERCEPTRON PROGRAM

```
        ORG 100
        HPR
        SWT 4                          TEST FOR NEW MATRIX
        TRA READ8
        SWT 3                          TEST FOR ZERO MATRIX
        TRA STIM-2
        TSX XINPUT,4
        NTR F3,,11
        MON MATRIX,,MATRIX+383
        TRA STIM-2
READ8   LXD INDEX1,1                   384
        RTB 8
        CPY MATRIX+384,1
        TIX *-1,1,1
        CLA HNDRED
        STO COUNT
STIM    TSX XINPUT,4
        NTR F1,,0
        MON SUNIT,,ANSWER+5
        REM COMPUTE RUNIT INPUTS AND OUTPUTS
        LXD INDEX1,1                   384
        LXD INDEX2,2                   6
NEXTR   LXA INDEX1,4                   64
        STZ RUNIT+6,2
        LDQ SUNIT+64,4
        MPY MATRIX+384,1
        ADD RUNIT+6,2
        STO RUNIT+6,2
        TXI *+1,1,-1
        TIX *-5,4,1
        LDQ ONE
        TPL *+2
        LDQ ZERO
        STQ RUNIT+6,2
        TIX NEXTR,2,1
        REM CHECK ANSWERS AND TRAIN
        LXD INDEX1,1                   384
        LXD INDEX2,2                   6
TESTR   CLA RUNIT+6,2
        SUB ANSWER+6,2
        TZE RIGHT
        SWT 1                          TEST FOR TRAINING
        TRA RIGHT-1
        LXA INDEX1,4                   64
        TMI INCRE
        CLA MATRIX+384,1
        SUB SUNIT+64,4
        STO MATRIX+384,1
        TXI *+1,1,-1
        TIX *-4,4,1
        TRA RIGHT-2
INCRE   CLA MATRIX+384,1
```

```
        ADD SUNIT+64,4
        STO MATRIX+384,1
        TXI *+1,1,-1
        TIX *-4,4,1
        TXI *+1,1,64
        CLA ONE
RIGHT   STO CHECK+6,2
        TXI *+1,1,-64
        TIX TESTR,2,1
        CLA ONE
        SWT 1                             TEST FOR TRAINING
        CLA ZERO
        STO TRAIN
        TSX OUTPUT,4
        NTR F2,,0
        MON SUNIT,,TRAIN
        CLA COUNT
        SUB ONE
        STO COUNT
        TZE *+3
        SWT 2                             TEST FOR STOP
        TRA STIM
        TSX OUTPUT,4                      PRINT PRESENT MATRIX
        NTR F4,,0
        MON MATRIX,,MATRIX+383
        SWT 2                             TEST FOR STOP
        TRA STIM-2
        LXD INDEX1,1                      384
        WTB 8
        CPY MATRIX+384,1
        TIX *-1,1,1
        HPR
        TSX RETURN,4
        HTR *-1
ZERO    OCT 000000000000
ONE     OCT 000001000000
HNDRED  OCT 000144000000                  100
INDEX1  OCT 000600000100                  384      64
INDEX2  OCT 000006000000                    6       0
F1      BCD 3(64I1,2X,6I1)
        OCT 777777777777
F2      BCD 5(10X,64I1,3(3X,6I1),3X,I1)
        OCT 777777777777
F3      BCD 1(8I6)
F4      BCD 1(16I6)
SUNIT   BSS 64
ANSWER  BSS 6
RUNIT   BSS 6
CHECK   BSS 6
TRAIN
COUNT
MATRIX  BSS 384
        END 100
```

APPENDIX 4 DATA GENERATION PROBLEM 1

```
      DIMENSION KLM(64)
      READ 100,Q
      DO 20 J=1,1000
      DO 10 K=1,64
      CALL NUMBER (Q,R)
 10   KLM(K)=R+.5
 20   WRITE OUTPUT TAPE 6,110,(KLM(K),K=1,64),(KLM(K),K=1,6)
      END FILE 6
 100  FORMAT (O12)
 110  FORMAT (64I1,2X,6I1)
       END
```

APPENDIX 5 DATA GENERATION PROBLEM 2

```
      DIMENSION M(64),N(6)
      DO 10 I=1,4
   10 N(I)=0
      L=2
      DO 60 I=1,1000
      READ INPUT TAPE 6,100,(M(J),J=1,64)
      IF (M(1)+M(2)+M(3)+M(4)+M(5)-M(6)-M(7)-M(8)-M(9)-M(10))20,20,30
   20 N(5)=0
      GO TO 40
   30 N(5)=1
   40 K=M(1)+M(2)+M(3)+M(4)+M(5)+M(6)
S     CLA K
S     TZE*50
S     SUB L
S     TZE*50
S     SUB L
S     TZE*50
S     SUB L
S     TZE*50
      N(6)=1
      GO TO 60
   50 N(6)=0
   60 WRITE OUTPUT TAPE 7,100,(M(J),J=1,64),(N(J),J=1,6)
      END FILE 7
      CALL RETURN
      STOP
  100 FORMAT (64I1,2X,6I1)
       END
```

DISTRIBUTION

	Copies
Assistant Secretary of Defense for Research and Development Information Office Library Branch Washington 25, D. C.	1
Chief, Bureau of Naval Weapons Washington 25, D. C.	
(RRRE)	1
(R-14)	1
(R-12)	1
(RU-231)	1
(RUDC-1)	1
(DLI-3)	1
Chief, Bureau of Ships Washington 25, D. C. Attn: (Technical Library) Code 631	1
Chief, Office of Naval Research Washington 25, D. C.	
(Code 411)	1
(Code 466)	1
Chief of Naval Operations Washington 25, D. C.	
(Op 311)	1
(Op 312)	1
(Op 316)	1
(Op 93EG)	1
(Op 93R)	1
Commander U. S. Naval Ordnance Test Station 3202 East Foothill Boulevard Pasadena, California	1
Commanding Officer U. S. Naval Underwater Ordnance Station Newport, Rhode Island	1
Commanding Officer U. S. Naval Air Development Center Johnsville, Pennsylvania	1

DISTRIBUTION (CONT.)

Copies

Commanding Officer and Director
U. S. Navy Electronics Laboratory
San Diego 52, California 1

Commanding Officer and Director
U. S. Navy Underwater Sound Laboratory
Fort Trumbull
New London, Connecticut 1

Commanding Officer and Director
David Taylor Model Basin
Washington 7, D. C. 1

Director
U. S. Navy Underwater Sound Reference Laboratory
Orlando, Florida 1

Commander
U. S. Naval Ordnance Test Station
China Lake, California 1

Commanding Officer
U. S. Mine Defense Laboratory
Panama City, Florida 1

Commanding Officer and Director
Naval Research Laboratory
Washington 25, D. C.
(Code 2021) 1
(Code 4000) 1

Director
Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 1

Director
Ordnance Research Laboratory
Pennsylvania State College
University Park, Pennsylvania 1

Director
Scripps Institution of Oceanography
University of California
La Jolla, California 1

DISTRIBUTION (CONT.)

	Copies
Director Applied Physics Laboratory University of Washington Seattle 5, Washington	1
Director Hudson Laboratory Columbia University Dobbs Ferry, New York	1
Director Marine Physical Laboratory San Diego, California	1
National Research Council Committee on Undersea Warfare Executive Secretary 2101 Constitution Avenue, N.W. Washington, D. C.	1
Cognitive Systems Group Cornell Aeronautical Laboratory Buffalo, New York Attention: F. Rosenblatt	1
Communications and Data Processing Division Raytheon Company Norwood, Massachusetts Attention: J. P. J. Gravell	1

Naval Ordnance Laboratory, White Oak, Md.
(NOL technical report 61-114)
THE PERCEPTRON AS AN ADAPTIVE CLASSIFICATION DEVICE (U), by C.N. Pryor, jr. 26 Oct. 1961. 18p. charts. Task RUSD-4C150. UNCLASSIFIED

Under certain conditions an adaptive data system can be built to determine a solution to the data classification problem, based on samples of input data along with their proper classification. The perceptron and the simpler half-perceptron are machines of this sort, and their principles of operation and simulation on an IBM 704 computer are discussed. Sample problems exhibit the capabililities and limitations of each. The possible role of these machines in sonar target classification is also discussed.

1. Learning machines
2. Perceptrons
3. Data - Reduction
4. Computers - IBM 704
I. Title
II. Pryor, C. Nicholas, jr.
III. Project

Naval Ordnance Laboratory, White Oak, Md.
(NOL technical report 61-114)
THE PERCEPTRON AS AN ADAPTIVE CLASSIFICATION DEVICE (U), by C.N. Pryor, jr. 26 Oct. 1961. 18p. charts. Task RUSD-4C150. UNCLASSIFIED

Under certain conditions an adaptive data system can be built to determine a solution to the data classification problem, based on samples of input data along with their proper classification. The perceptron and the simpler half-perceptron are machines of this sort, and their principles of operation and simulation on an IBM 704 computer are discussed. Sample problems exhibit the capabililities and limitations of each. The possible role of these machines in sonar target classification is also discussed.

1. Learning machines
2. Perceptrons
3. Data - Reduction
4. Computers - IBM 704
I. Title
II. Pryor, C. Nicholas, jr.
III. Project

Naval Ordnance Laboratory, White Oak, Md.
(NOL technical report 61-114)
THE PERCEPTRON AS AN ADAPTIVE CLASSIFICATION DEVICE (U), by C.N. Pryor, jr. 26 Oct. 1961. 18p. charts. Task RUSD-4C150. UNCLASSIFIED

Under certain conditions an adaptive data system can be built to determine a solution to the data classification problem, based on samples of input data along with their proper classification. The perceptron and the simpler half-perceptron are machines of this sort, and their principles of operation and simulation on an IBM 704 computer are discussed. Sample problems exhibit the capabililities and limitations of each. The possible role of these machines in sonar target classification is also discussed.

1. Learning machines
2. Perceptrons
3. Data - Reduction
4. Computers - IBM 704
I. Title
II. Pryor, C. Nicholas, jr.
III. Project

Naval Ordnance Laboratory, White Oak, Md.
(NOL technical report 61-114)
THE PERCEPTRON AS AN ADAPTIVE CLASSIFICATION DEVICE (U), by C.N. Pryor, jr. 26 Oct. 1961. 18p. charts. Task RUSD-4C150. UNCLASSIFIED

Under certain conditions an adaptive data system can be built to determine a solution to the data classification problem, based on samples of input data along with their proper classification. The perceptron and the simpler half-perceptron are machines of this sort, and their principles of operation and simulation on an IBM 704 computer are discussed. Sample problems exhibit the capabililities and limitations of each. The possible role of these machines in sonar target classification is also discussed.

1. Learning machines
2. Perceptrons
3. Data - Reduction
4. Computers - IBM 704
I. Title
II. Pryor, C. Nicholas, jr.
III. Project